Serverless Development 101

**HOP07 – Build an app with REST API - CREATE**

5/5/2020 Created by Apiwat Chuaphan

Center for Information Assurance (CIAE) @City University of Seattle (CityU)

  

**Learning Outcomes**

* Learn what REST API is and how it works
* Learn basic CRUD
* Learn how items store in DynamoDB
* Build an app with AWS Lambda, API Gateway, DynamoDB, REST API

A screenshot of a cell phone

Description automatically generated

**REST**

is acronym for **RE**presentational **S**tate **T**ransfer. It is architectural style for distributed hypermedia systems that defines a set of constraints to be used for creating Web services, making it easier for systems to communicate with each other.

REST requires that a client make a request to the server in order to retrieve or modify data and it consists of:

* an *HTTP* verb, which defines what kind of operation to perform

There are 4 basic HTTP verbs we use in requests to interact with resources in a REST system:

* + GET — retrieve a specific resource (by id) or a collection of resources
  + POST — create a new resource
  + PUT — update a specific resource (by id)
  + DELETE — remove a specific resource by id
* a *header*, which allows the client to pass along information about the request
* a *path* to a resource
* an optional message body containing data

**Principles of CRUD**

The principles of the CRUD cycle are defined as CREATE, READ, UPDATE, and DELETE. These form the standard database commands. CRUD often corresponds to the HTTP methods POST, GET, PUT, and DELETE, respectively. These are the fundamental elements of a persistent storage system.

The Differences

* REST is an architectural system centered around resources and hypermedia, via HTTP protocols
* CRUD is a cycle meant for maintaining permanent records in a database setting
* CRUD principles are mapped to REST commands to comply with the goals of RESTful architecture

**Build the C from CRUD paradigm.**

1. Open the VSCode and open the “**myproject**” project folder that we developed a web application using serverless computing in the previous module.
2. Run **serverless create --template aws-nodejs --path todos** to create another folder name “**todos**” under “**myproject**” folder.
3. Create **package.json** file under **todos** folder with the following content [here](https://bit.ly/2X1SCb0)
4. Run **npm install** in the terminal to install dependencies. Note that **node\_modules** folder has been created.
5. Replace **serverless.yml** with the content from [here](https://bit.ly/2X5rsQi)

A screenshot of a cell phone

Description automatically generated

This will allow Lambda to have access to DynamoDB with the methods shown.

1. Create new folder named “**functions**” under “**todos**”
2. Create a file named **create.js** under **functions** folder with the content [here](https://bit.ly/3fZO8KO)

A screenshot of a cell phone

Description automatically generated

At first, we'll require all the necessary packages we need. We'll require the AWS SDK, create a new DyanmoDB instance with the help of the AWS SDK and require the uuid package so that we can generate unique ids.

***AWS.DynamoDB.DocumentClient()*** - The document client abstraction makes it easier to read and write data to Amazon DynamoDB. This [answer](https://stackoverflow.com/questions/57804745/difference-between-aws-sdk-dynamodb-client-and-documentclient) might be helpful.

Inside of this function we'll get the data of the todo we will create from the event parameter then it will be passed from API Gateway down to the Lambda function.

Then call DynamoDB and store the new data in table. At the end we call the callback and return either the success or an error.

1. Run **sls deploy -v** to deploy your app to Lambda with -v option (verbose mode) to print the progress during the deployment.

A screenshot of text

Description automatically generated

*blah*

*blah*

*blah*

A screenshot of a cell phone

Description automatically generated

1. Head to AWS Lambda page to see there is new function deployed.

A screenshot of a social media post

Description automatically generated

If you made some changes to a function and want to deploy again, use **sls deploy function -f <function name>** for faster deploy.

1. Test our **create.js** API by typing this command in the terminal

**>>> curl -X POST -d '{"text":"finish this HOP"}' <your-url>**

**curl** is a command line tool for transferring data via URLs.

**-X, --request -** Custom request method

**-d, --data -** Sends the specified data

**-H, --header -** Sends headers

**-i, --include -** Display response headers

The return result in the terminal looks something like this.



If you prefer to use GUI tool, Postman is a great option for doing so. [Download here](https://www.postman.com/)

A screenshot of a cell phone

Description automatically generated

1. Now, go to AWS DynamoDB page to see if new todo data has been added.

­­­A screenshot of a cell phone

Description automatically generated

Once you click on the item, you will see that our todo data has been added to the database.

A screenshot of a cell phone

Description automatically generated

We’ll read data from the database in the next module. I know you’re excited, I’m too. 😀

**Push your work to GitHub**

Open the terminal from the VSCode by hit the control + ~ key and type the following command:

Run the following commands to push your work to the GitHub repository:

>>> git add .

>>> git commit -m “Submission for Module 7”

>>> git push origin YOUR\_BRANCH\_NAME

**Note**: you should change the YOUR\_BRANCH\_NAME to your own branch name. It should be firstname-lastname (e.g. maria-gracia).

If you cannot remember, run the command “git status” to check